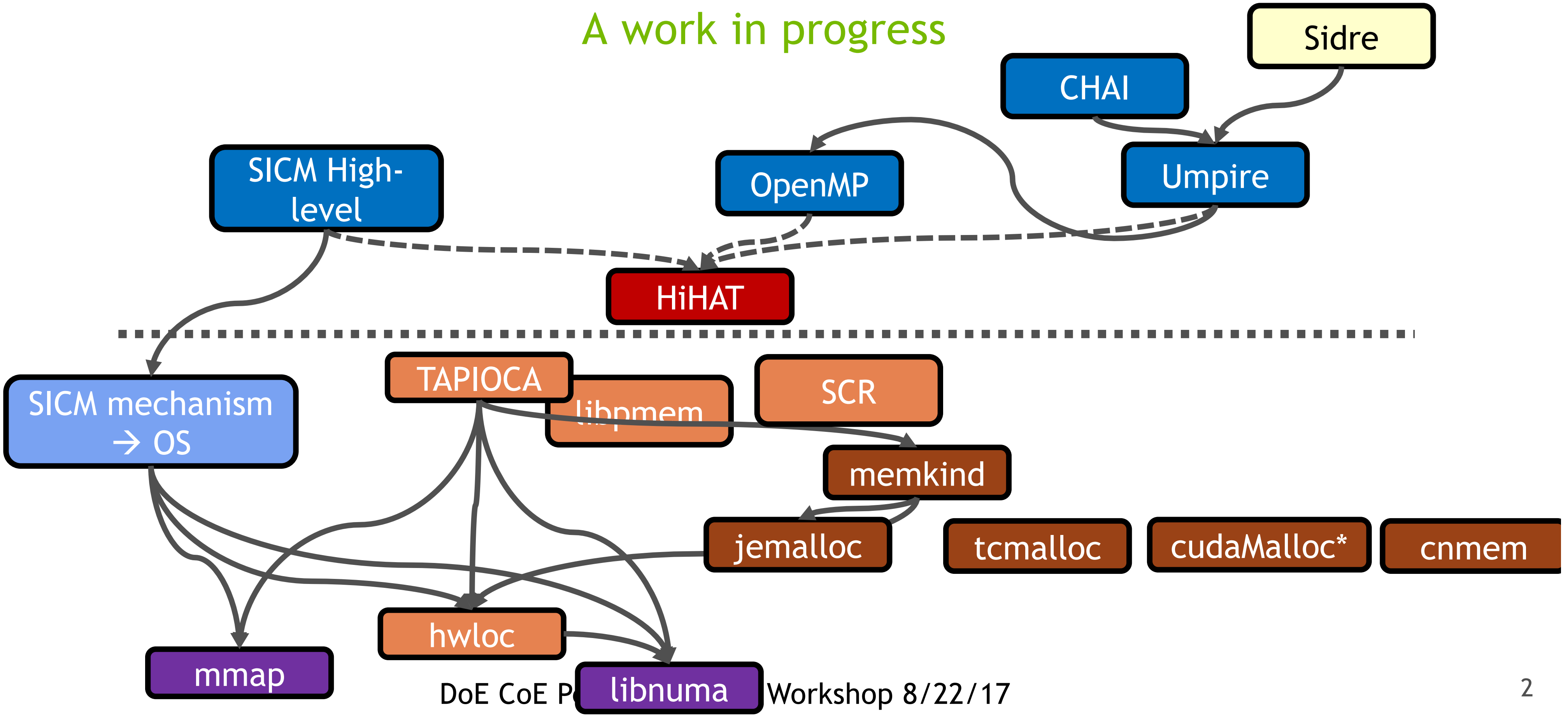
MEMORY ABSTRACTIONS BREAKOUT

- Moderators: Ian Karlin - LLNL, CJ Newburn – NVIDIA
- ~55 participants

MEMORY ABSTRACTIONS - WIP

A work in progress



PANEL QUESTIONS

- **where your work fits** in the diagram
- where we think there's **agreement**
  - abstraction, let people have their own customized interfaces, C ABI, traits
  - collaboration, leverage arch features, driven by app usage models
- where we think there are **opens**
  - How to do layering, collaboration, common abstractions; moving toward standards
  - What traits and which are mutable
  - Cost models and enumeration: unknown vendor info, what to enumerate
  - Emerging ideas: asynchrony
  - Memory and storage: common abstraction, floating point precision to save/restore, policies, abstractions for data layout, gap between storage and memory (mmap & files)
  - Introspection: what to reveal, how to deal with dynamism
- how we think we should **move toward closure** of those opens
  - share requirements, proofs of convept, work with vendors

VENDOR-SPECIFIC SOLUTIONS FOR MANAGING MEMORY

Do people actually need this? Is it actually worthwhile?

Want sensible portable defaults first, then add knobs.

We solved this problem with UVM. That made codes port beautifully to GPU.

CUDA UVM made sparse linear solvers work in Trilinos. Must be careful of performance issues.

Strategy: CUDA UVM everywhere, get code running, then profile and reduce data motion. Just starting with managed memory was hard.

FUTURE OF MULTI-LEVEL MEMORY

Differentiation between vendors and apps

Apps people have spoken its too hard to use

- Strong scale to fit in fast memory
- We should ask for a flatter space

Not getting MLM systems will limit total memory footprint

Could be the abstractions are wrong

- KISS, but it didn’t work with HPF you ended up writing MPI like code

INTEROPERABILITY (COMPLETELY MISSED)

- All of these ideas need to work together in multi-physics codes
- Each library or physics package is choosing its own approach
- How are pointers passed between packages

WRAP-UP

- Do we want to see clear distillation of requirements and available options? -
- If so, are you willing to contribute to efforts of expressing those requirements? -
- If so, are you willing to work towards sharing code and infrastructure towards layering?
- Easy to agree there's a problem; hard to act on this.
- Too many implementations is a huge waste of productivity.

DISCUSSION

- Can users define classes of memory, based on their own notion of which traits are important and what the enumerated or characterized values are?
  - Kokkos does something like this
- Allocators – in C++ standard
- Abstraction for data layout – Kokkos template, Anshu & Fortran
- Cost models – Dmitry: for allocation; CJ: count costs of asynchrony, throughput vs. latency
- Policies – eviction
- Requirements: introspection
- Collaboration: CJ: open framework to compare/constrast code models & schedulers

ADDITIONAL TOPICS

- **Audience reactions** on the above topics and panelist positions
- Discuss what has been done and needs to be done to make solutions **target multiple architectures**
  - Rich: Retargetability: still need to reason, regardless of specifics
  - Livermore: Using managed memory by default via jemalloc was hard (libibverbs alloc/free mismatch)
  - Unified memory solved a deep copy problem
  - Trilinos wouldn't have been successful on GPUs without unified memory; brought up perf issues that needed to be reasoned about, need for fences

ADDITIONAL TOPICS

- Examine **disruptions from technology trends**, such as the increasing bandwidth gap between HBM and DDR, or what happens when NVM is added to the mix.
  - Ian: What does it take to achieve locality? Boosted locality > changed algo > production
  - Mike Lang: Mix of simple and complex hierarchies
  - Ian: Need clear language in RFPs to get what you want
  - CJ: some apps will need dynamic scheduling, async allocation, deferred binding
  - Si: keep flat where we can, add prefetching
- Interoperability – Adam @ Livermore
  - Alloc/Free mismatch problem between allocators

DETAILS

- Comparison tables
- Panelist slides
- Additional notes

MEMORY ABSTRACTIONS - WIP

Interface	Abstracts implementations	Policies	User facing
OpenMP	Alloc, pinning	Declare	Alloc, resources, access
CHAI	Pointers		Caching, layout?
Sidre			Layout?, sharing
Umpire	Alloc, enum	Implement	Alloc, move
SICM			
memkind	Alloc		
HiHAT	Alloc, enum, layout,access, resources/sharing	Declare	

MEMORY IMPLEMENTATIONS - WIP

Interface	Component	Ubiquity	Primary function
mmap	OS	Standardized	Alloc, affinity
libnuma, numactl, mbind	Library	Standardized	Affinity
hwloc	Library	Common	Enumeration
TAPIOCA	Library	Research	Block abstraction, move
libpmem	Library	Common	Alloc, access for persistent mem
allocators	library	Variety	Alloc
Scalable Checkpoint Restart	Library	Production	Manage burst buffers

OpenMP 5 Memory Mgmt. (Breakout Slides)

2017 CoE Performance Portability Meeting

Denver, Colorado, USA

August 21, 2017

Stephen Olivier

Center for Computing Research, SNL-NM

Exceptional service in the national interest

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2017-9053 PE

Outline of the OpenMP Approach

- Core features (near completion)
  - OpenMP allocators
  - `allocate` directive and clause
  - `omp_alloc()` and `omp_free()` API routines for C/C++
  - Default allocator
  - `declare alloc` directive and `omp_alloc.h`

- Additional features (less mature)
  - Memory spaces and traits
  - User-defined custom allocators
  - Fallback options
  - Other additions: better C++ and NUMA support, querying, etc.

Agreed?

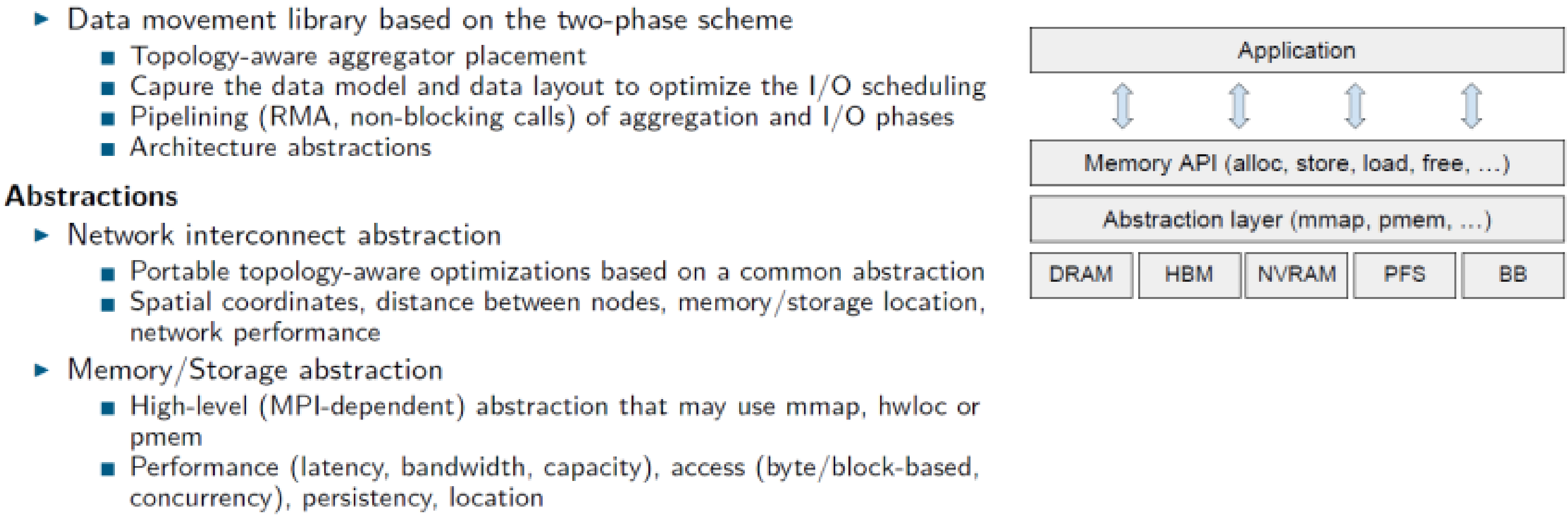
- Performance portability should involve the vendors getting together with the the users
  - Only vendors can choose what to share of their roadmaps, where memory strategies can changing dramatically and rapidly
  - User perspective on the performance-portability tradeoff
- Not possible to be completely future-proof, but consider how future architectures can be supported
  - Need something soon, but don’t want to orphan the code later
- Portability not just between vendors, but also languages
  - Fortran needs support too, and nice if it looks similar
- Vendor implementations needed
  - GCC and LLVM are good too, but we need supported compilers

Open?

- Should we have users describe desired properties for placement of their data *OR* tell the system exactly where to put data?
  - Again, performance-portability tradeoff
- How important is a decent standards-based solution?
  - Standards easier to include in procurements than the hard-to-define concept of “performance portability”
  - Probably won’t be the best on any particular architecture but do something reasonable on many architectures

TAPIOCA

Francois **Tessier**



- Data movement library based on the two-phase scheme
  - Topology-aware aggregator placement
  - Capure the data model and data layout to optimize the I/O scheduling
  - Pipelining (RMA, non-blocking calls) of aggregation and I/O phases
  - Architecture abstractions

Abstractions

- Network interconnect abstraction
  - Portable topology-aware optimizations based on a common abstraction
  - Spatial coordinates, distance between nodes, memory/storage location, network performance
- Memory/Storage abstraction
  - High-level (MPI-dependent) abstraction that may use mmap, hwloc or pmem
  - Performance (latency, bandwidth, capacity), access (byte/block-based, concurrency), persistency, location

Application

Memory API (alloc, store, load, free, ...)

Abstraction layer (mmap, pmem, ...)

DRAM | HBM | NVRAM | PFS | BB

SICM (SIMPLIFIED INTERFACE TO COMPLEX MEMORY)

Mike Lang

- Two level library implementation + kernel mods
  - High level application interface
  - Lower level mechanism interface
  - Kernel modifications (additional memory policy and multiple NUMA orders)
- Collaborators (GaTech, LLNL, ORNL) looking at higher-level interfaces, making decisions for higher-level apps
- Fleshed out some of what was missing in memkind across different archs
- Policies
  - How you spill
  - Seeking to push minor mods to the linux kernel

UMPIRE, RAJA, CHAI, SIDRE

David Beckingsdale

- Sidre
    - mesh and field data with schema that enables interop, used @ Livermore
    - Use attributes for data
- Agreements
    - Let people have their own interface flavor
- Opens
    - What do you expose from HW

SCALABLE CHECKPOINT RESTART (SCR)

Elsa Gonsiorowski, LLLNL

- Fit
  - Part of storage hierarchy, efficient path from memory to files
- Agreements
  - Similar strategies as for memory abstraction, but code not shared
- Opens
  - Gap where storage meets memory. mmap doesn’t handle need for accessing files.
  - Floating point precision that needs to be stored

OPENMP AND MEMORY CONSTRAINTS

Tom Scogland

- Fit
  - Just above mmap or malloc or just below user
- Agreements
  - Allocate and deallocate (oop, garbage collection)
  - Vertical and horizontal placement
- Opens
  - Bottom-level target-specific interfaces; “X won’t work with AMD GPU, Y won’t work with NVIDIA…"
- Closure

HIHAT

CJ Newburn

- **where your work fits** in the diagram
- where we think there’s **agreement**
  - abstraction, collaboration, leverage arch features, multi-language (C ABI)
- where we think there are **opens**
  - layering, collaboration, async, traits, mutability, standards, unknown vendor info, what to enumerate
- how we think we should **move toward closure** of those opens
  - share requirements, PoCs, vendor impl

DISCUSSION

- Can users define classes of memory, based on their own notion of which traits are important and what the enumerated or characterized values are?
  - Kokkos does something like this
- Allocators – in C++ standard
- Abstraction for data layout – Kokkos template, Anshu & Fortran
- Cost models – Dmitry: for allocation; CJ: count costs of asynchrony, throughput vs. latency
- Policies – eviction
- Requirements: introspection
- Collaboration: CJ: open framework to compare/constrast code models & schedulers

ADDITIONAL TOPICS

- **Audience reactions** on the above topics and panelist positions
- Discuss what has been done and needs to be done to make solutions **target multiple architectures**
  - Rich: Retargetability: still need to reason, regardless of specifics
  - Livermore: Using managed memory by default via jemalloc was hard (libibverbs alloc/free mismatch)
  - Unified memory solved a deep copy problem
  - Trilinos wouldn’t have been successful on GPUs without unified memory; brought up perf issues that needed to be reasoned about, need for fences

ADDITIONAL TOPICS

- Examine **disruptions from technology trends**, such as the increasing bandwidth gap between HBM and DDR, or what happens when NVM is added to the mix.
    - Ian: What does it take to achieve locality? Boosted locality > changed algo > production
    - Mike Lang: Mix of simple and complex hierarchies
    - Ian: Need clear language in RFPs to get what you want
    - CJ: some apps will need dynamic scheduling, async allocation, deferred binding
    - Si: keep flat where we can, add prefetching
- Interoperability – Adam @ Livermore
    - Alloc/Free mismatch problem between allocators